SERVIÇO PÚBLICO FEDERAL
UNIVERSIDADE FEDERAL DE SERGIPE
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA

PROGRAMA DE INICIAÇÃO CIENTÍFICA VOLUNTÁRIA – PICVOL

INVESTIGANDO A PLATAFORMA FIWARE NO DESENVOLVIMENTO DE APLICAÇÕES PARA CIDADES INTELIGENTES

Área do conhecimento: Ciências Exatas
Subárea do conhecimento: Ciência da computação
Especialidade do conhecimento: IoT, Smart Cities, Open Data

Relatório Final
Período da bolsa: de agosto 2017 a julho 2018

Este projeto é desenvolvido com bolsa de iniciação científica
PICVOL

Orientador: Dra. Leila Maciel de Almeida e Silva
Autor: Weslan Rezende Alves

SUMÁRIO

1. INTRODUÇÃO

O aumento populacional no planeta foi exorbitantemente, pesquisas revelam que em pouco mais de dois séculos a população setuplicou em todo o mundo (Department of Economic and Social Affairs, 2017).

Além disso, pesquisas também comprovaram que a maioria dos cidadãos vivem em áreas urbanas, só no Brasil em 2010 foi revelado que 84,36% da população mora em zona urbana e apenas 15,64% na rural (Urbe Lab, 2015).

Com uma quantidade enorme de cidadãos a complexidade para gerir uma cidade cresce em paralelo ao número da população, assim cerca de 10 anos atrás (Bayer Jovens, 2016) surgiu o conceito de cidades inteligentes (*Smart City*); cujo objetivo é gerenciar o aumento populacional de forma sustentável em prol da melhor qualidade de vida. A aplicação deste conceito é feita através do uso intensivo da tecnologia da informação e comunicação.

Para tornar um ambiente inteligente é necessário primeiro torná-lo um laboratório, dando toda infraestrutura para pesquisas e projetos. Os projetos são direcionados nas seguintes áreas: Internet das Coisas (*Internet of Things* - IoT), dados volumosos (*Big Data*) e Governança Algorítmica. Desta forma são realizados experimentos em pilares da sociedade como: educação, saúde, esporte, infraestrutura, etc.

Toda vez que um projeto é realizado, ele deve ter como objetivo minimizar custos governamentais ou introduzir melhorias de processos. A área de saúde pública no Brasil é muito custosa e com pouca eficácia no atendimento.Assim, no conceito de cidades inteligentes, surgiram algumas tentativas para tornar o atendimento público de saúde mais eficaz, como por exemplo:

- e-SAÚDE: um aplicativo desenvolvido pelo Departamento de Informática do SUS – DATASUS, cujo objetivo é aproximar o cidadão das unidades de saúde público do estado. Ele é de uso pessoal e ao usar o aplicativo o usuário tem acesso a informações e pode também inserir informações pessoais no aplicativo (Silvano Vilela, sem data).
- e-SUS AB Território (DATASUS, 2017): este aplicativo é objetiva auxiliar os agentes de atendimento domiciliar em geral. Nele é possível cadastrar prontuários previamente configurados pelo município, o aplicativo também faz uma integração dos dados com o Prontuário Eletrônico do Cidadão (PEC). Desenvolvido pelo departamento de Informática do SUS – DATASUS.
- Viva Bem (DATASUS, 2017): assim como o e-SAÚDE o aplicativo Viva Bem é acessado de forma individual, entretanto seu objetivo é fazer com que o usuário possa se organizar junto ao município. Este

> aplicativo disponibiliza uma série de funcionalidades como: cadastro de lembretes de medicamentos, visualização das informações do medicamento, resultado de exames, etc., tornando o acesso a informações médicas mais próxima dos cidadãos. Desenvolvido pelo departamento de Informática do SUS – DATASUS.

Note que todos esses projetos têm como objetivo a melhoria de processos, o que os classificam como um projeto de cidade inteligente, no entanto é notório ressaltar que nenhum dos aplicativos criados pelo departamento de Informática do SUS (DATASUS) tem como objetivo auxiliar no combate ao mosquito *Aedes Aegypti.*

O combate aos mosquitos transmissores epidêmicos é um processo contínuo e, embora estatísticas aponte que casos de dengue, zika e chikungunya estejam caindo (Combate aedes, 2017), o número de casos ainda é alto e não se tem um controle adequado dos dados coletados sobre as incidências das doenças.

Embora os dados apontem que até 15 de abril de 2017, tenha ocorrido uma queda de 90,3% (Combate aedes, 2017) dos casos em relação ao mesmo período do ano anterior (2016), o número de casos ainda é assustador em um curto período de tempo.

Neste contexto, vislumbra-se a necessidade de informatizar e melhorar a forma de acesso aos dados decorrentes do controle do Aedes Aegypti. Usaremos, para isto, a plataforma FIWARE, ao qual é desenvolvida para dar suporte a projetos de cidades inteligentes, objetivando adquirir conhecimentos sobre a mesma para que possamos elaborar e desenvolver uma aplicação interoperável para auxiliar os agentes comunitários de saúde no combate ao mosquito Aedes Aegypti.

2. OBJETIVO

Estudar amplamente a plataforma FIWARE e desenvolver e testar uma aplicação que auxilie a notificação de focos do Aedes Aegypti.

3. ATIVIDADES REALIZADAS

Na primeira etapa da iniciação científica fizemos uma revisão bibliográfica na área de cidades inteligentes, identificamos os principais conceitos e tópicos relacionados ao tema. Em especial foi estudado a área de saúde coletiva, investigando como uma cidade pode melhorar o atendimento público através da tecnologia da informação.

Paralelo às pesquisas estudamos também a plataforma FIWARE (FIWARE Developers), com foco no componente Orion Context Broker. Esta plataforma é largamente utilizada para desenvolver aplicações de cidades inteligentes.

Estudamos amplamente como instalar e configurar a plataforma, como usar e desenvolver utilizando suas funcionalidades.

Para o desenvolvimento da aplicação, consultamos a Dra. Roseli La Corte dos Santos, do departamento de Morfologia da UFS, para colher informações do que um agente epidemiológico necessita no dia-a-dia do combate ao mosquito Aedes Aegypti, como formulários, hierarquia dos agentes, função de cada membro e atividades desempenhadas. A partir da entrevista com a profa. Roseli especificamos, projetamos e desenvolvemos a aplicação de monitoramento do Aedes Aegypti, no contexto do FIWARE. A aplicação consiste em dois módulos, o mobile e o web, onde a parte mobile é responsável por coletar e trabalhar junto aos agentes de saúde, enquanto a web controla os dados e supervisiona as equipes de saúde.

4. JUSTIFICATIVA DE ALTERAÇÃO NO PLANO DE TRABALHO

Não houve alteração no plano de trabalho. No entanto houve um atraso na conclusão do fundamento devido à dificuldade de contatar a Secretária de Saúde, que foi a nossa primeira tentativa para coletar dados sobre a rotina dos agentes de saúde. Assim a validação da ferramenta ficou prejudicada.

5. OUTRAS ATIVIDADES

Além das pesquisas bibliográficas, participamos também do evento Hackathon Carmelita; primeira maratona cívica de programação realizada em Sergipe com o apoio da UFS e a prefeitura de São cristóvão. Neste evento tivemos a oportunidade de aprender um pouco mais, através de palestra, sobre como tornar uma cidade inteligente (Hackathon Carmelita, 2017).

No mesmo, aplicamos nossos conhecimentos da plataforma FIWARE em uma aplicação de geo-mapeamento de famílias em zonas de risco, aplicação esta que nos levou ao 1º (primeiro) lugar no evento.

Já na segunda etapa do PIBIC, por meio dos nossos estudos, tive a oportunidade de palestrar falando sobre o FIWARE para o evento Hackathon UFS II que ocorreu no Hospital Universitário, nos dias 15,16 e 17 de junho.

6. REVISÃO DA LITERATURA

O foco da nossa proposta é o desenvolvimento da ferramenta de monitoramento utilizando o FIWARE.

Assim, concentramos nossos esforços na descrição do FIWARE.

Antes de falarmos sobre o FIWARE, é preciso entender o conceito sobre uma API - *Application Programming Interface* (Interface de Programação de Aplicações). Atualmente, é comum que as aplicações (cliente e servidor) necessitem de uma comunicação por meio da Internet. Sendo assim, as aplicações do servidor são desenvolvidas com rotinas e funcionalidade para que as aplicações de cliente (independente da tecnologia usada), possam realizar funcionalidades da aplicação do servidor sem precisar saber dos detalhes.

Sendo assim, as aplicações do servidor são conhecidas como API's e disponibilizam um canal ou interoperabilidade entre *softwares*.

Neste contexto REST - *Representational State Transfer* (Transferência de Estado Representacional). É um termo que surgiu do protocolo HTTP, logo após o

ano 2000, com o surgimento de novos métodos de comunicação: GET, DELETE, POST (BECODE, sem data). Foi a criação destes novos métodos que possibilitou a padronização de API's. Sendo assim, uma API REST trata-se da abstração da arquitetura *Web* com princípios e regras, que permitem a criação de projetos bem definidos.

6.1. HTTP 1.1

HyperText Transfer Protocol (Protocolo de transferência de Hipertexto) é um protocolo de comunicação utilizado para a transferência de dados entre sistemas. Atualmente este protocolo é comumente utilizado em sites, aplicações *Web* e requisições de aplicações. Por isto, ao acessar um site necessitamos informar a *Uniform Resource Locator* (URL) para que o servidor possa acessar e retornar a página solicitada. Sendo assim, o HTTP consiste em padronizar as requisições e resposta. Atualmente possui os seguintes métodos:

- GET: utilizada para requisitar um recurso que pode ser um arquivo, com por exemplo: *HyperText Markup Language* (HTML), J*avaScript Object Notation* (JSON), *Extensible Markup Language* (XML), PHP, dente outros;
- POST: utilizado para criar um recurso. Diferente do GET, o POST armazena os dados no corpo da requisição;
- PUT: solicita o armazenamento/atualização de uma única parte do recurso;
- DELETE: solicita a exclusão de um recurso;
- PATCH: assim como o PUT, serve para armazenar ou atualizar um recurso. No entanto, no PATCH é feito para mudar mais de uma parte do recurso;
- TRACE: envia uma mensagem de *loopback* até o destino da requisição provendo uma forma de *debug*, se o destino receber a mensagem, ele excluirá alguns campos e retornará a mesma requisição;
- OPTIONS: informa os métodos suportados pelo destino da requisição;
- CONNECT: transforma a requisição para um TCP/IP transparente, esse método é feito para obter maior segurança em *proxys* não criptografados;
- HEAD: realiza uma requisição que retorna somente os cabeçalhos da resposta do destino.

Para este documento iremos necessitar apenas dos métodos GET, POST, DELETE, PUT e PATCHO.

6.2. COMPARTILHAMENTO DE RECURSOS ENTRE ORIGENS (CROSS-ORIGIN RESOURCE SHARING - CORS)

Atualmente os navegadores fazem uso da política de segurança de mesma origem (*Same-origin Policy*), em que um site só pode fazer requisições a outro site, se e somente se, os sites estiverem no mesmo domínios.

Essa norma dos navegadores não conseguiu acompanhar os avanços tecnológicos do HTML5 que, com a nova versão, permite que um site possa acessar recursos de outro site, mesmo estando em domínios diferentes.

Devido à política de segurança dos navegadores, API REST feitas por JavaScripts tornam-se limitadas ao acesso às novas tecnologias do HTML5, pois esta linguagem roda ao lado do cliente e necessita de uma interpretação do browser (navegador) para que a ação seja executada. Desta forma, as API limitavam-se somente a utilização do método GET, o que contradiz o conceito de REST.

Para contornar essa situação, foi desenvolvido o *Cross-Origin Resourcer Sharing* (CORS), permite que a comunicação entre domínios seja possível em qualquer método (GET, POST, DELETE, PUT, PATCH, dentre outros), desde que, o domínio destino especifique este tipo de comunicação. Sendo assim, ficou por responsabilidade das API's especificarem nos cabeçalhos de suas respostas o tipo de comunicação: "*Access-Control-Allow-Origin*".

Veremos que o FIWARE contém uma API Orion a qual normalmente não fica em mesmo domínio das aplicações que irão utilizá-la. Dessa forma, apresentamos no Anexo I que ao instalar a API Orion devemos então configurar o cabeçalho da API para incluir o CORS (por meio de comandos), pois por padrão o FIWARE não põe cabeçalhos de permissão de acesso de origem diferentes.

6.3. MIDDLEWARE

O desenvolvimento de *software* nunca teve tanta força quanto nos últimos anos, conhecidos como era da tecnologia, sendo assim cada vez mais é investido em aplicações inteligentes.

Graças à expansão das Tecnologias da Informação, o termo Cidades Inteligentes (CI) nunca havia sido tão citado entre governantes. Estes estão buscando meios tecnológicos para gerir gastos, coleta de lixo, saúde pública, dentre outros. Mas para obter tamanha inteligência e capacidade de tomada de decisão, os *softwares* tornaram-se complexos, o que tornou o desenvolvimento cada vez mais demorado.

Para solucionar essa *déficit* de desenvolvimento, começaram a surgir plataformas de *middleware* (plataformas mediadoras) que tentam tornar o

desenvolvimento mais ágil e fácil tornando transparente as camadas de desenvolvimento. Além disso, torna as aplicações interoperáveis e capacita a coleta de dados volumosos.

Chamamos de *middleware* toda plataforma que está entre dispositivos e bases de dados, por exemplo: em uma empresa existem diversos setores que necessitam de insumos, para solicitar estes insumos eles buscam o almoxarife, que por sua vez, vai até o estoque e entrega o insumo solicitado. Neste exemplo o almoxarife é uma espécie de "*middleware*", pois para os setores necessitados basta requisitar um material sem se preocuparem com a complexidade de gerir um estoque, armazenar os insumos de forma lógica para facilitar as buscas, registrar as quantidades de material em estoque, dentre outros. Ou seja, um *middleware* retira complexidades das aplicações.

Sendo assim, tais plataformas são desenvolvidas para abstrair camada por camada de um *software* (se assim for necessário) tornando o desenvolvimento claro, estruturado e padronizado possibilitando uma maior escalabilidade.

No entanto, ainda não se encontrou um padrão específicos para esses mediadores, devido à recente demanda para esses *softwares*.

6.4. MIDDLEWARE *PARA CIDADES INTELIGENTES*

Há um grande desafio para desenvolver um *middleware* para Cidades Inteligentes (CI). No entanto, estes possuem um nível de complexidade, que necessitam atingir alguns pré-requisitos para serem considerados *middleware* inteligente. Segundo Pires (2016), os requisitos para uma plataforma inteligente são: interoperabilidade; descoberta e gerenciamento de dispositivos; adaptação dinâmica; ciência de contexto; escalabilidade; tratamento de grandes volumes de dados; segurança; gerenciamento de dados; e ferramentas para o desenvolvimento de aplicações.

Além desses requisitos é fundamental que uma cidade, verdadeiramente inteligente, tenha meios de disponibilização de dados abertos, pois estes dados podem servir para análise (para exibição), extração de resultados a partir dos mesmos dados e até mesmo tomada de decisão de acordo com a análise realizada. Sendo assim, um *middleware* não pode se esquecer de viabilizar os portais OPEN DATA.

6.4.1. INTEROPERABILIDADE

Interoperabilidade é capacidade que um sistema possui de se comunicar com outros sistemas de forma transparente, para que um sistema seja interoperável é indispensável que ele use padrões abertos.

Um sistema não é considerado somente pela capacidade de transmitir dados entre sistemas, na verdade é necessário que ele possa se comunicar com o meio, por meio de ações, dados e que possa deixar um legado para outros aplicativos.

Sendo assim é indispensável que um *middleware* tenha interoperabilidade entre suas camadas e possibilite que os *softwares* que use a plataforma possam ser interoperáveis tanto em *hardware*, *software*, protocolos, formatos de dados, etc. Segundo Batista et. al. (2016, p.7), diz que:

> "... a integração de dispositivos atinge diversos níveis, desde (i) um mais baixo nível, no qual é necessário integrar os dispositivos físicos de maneira transparente e ocultar os detalhes como relação à rede, formatos de dados empregados, e até mesmo à semântica das informações, passando por (ii) um nível intermediário, no qual é necessário integrar e disponibilizar dados providos pelos dispositivos, de forma a prover serviços de valor agregado aos usuários, até, por fim, (iii) um mais alto nível, no qual a agregação e transformação das informações dos dispositivos são providas por um modelo padronizado de programação, de forma a permitir que desenvolvedores de aplicações não necessitem ter o conhecimento acerca das especificidades dos dispositivos físicos e do ambiente de rede subjacente."

Sendo assim, é de suma importância que a plataforma possibilite a integração de sistemas, pois as aplicações são distintas, com tecnologias diferente e vários objetivos, portanto ser interoperável é indispensável para uma *middleware* que deseja atender CI.

6.4.2. DESCOBERTA E GERENCIAMENTO DE DISPOSITIVOS

É comum que existam diversos dispositivos em um meio (sensores, celulares, câmeras, dentre outros), torna-se então necessário a capacidade de gerir estes dispositivos.

Um dos desafios dessa gestão é a instabilidade dos dispositivos, pois os mesmo podem entrar ou sair do ambiente a qualquer momento, seja por instabilidade da rede de Internet, erros ou até mesmo de forma intencional. Sendo assim, um *middleware* deve gerir de forma dinâmica a comunicação entre os

sistemas, endereçando ou dando cabeçalhos, de forma que haja uma organização e disponha informações como localização ou o estado do dispositivo.

6.4.3. ADAPTAÇÃO DINÂMICA

A capacidade de adaptar-se ao meio é de extrema importância para um *middleware*, tendo em vista que o meio não contém padrões e é dinâmico a plataforma necessita de recursos para adaptar-se ou reconfigurar-se, de forma a prover uma alta qualidade, aos dispositivos que ela dá suporte, e disponibilidade.

Essa dinamicidade deve ser sucinta de forma que não gere erro ou inconsistência no sistema.

Este requisito é de suma importância para sistema considerados críticos, por exemplo, aeronaves que sofrem pane de software devem ter a capacidade de identificar o erro e corrigi-los para que seja evitado catástrofes que podem levar perdas de vidas humanas.

6.4.4. CIÊNCIA DE CONTEXTO

Em um ambiente conectado, um sensor pode gerar diversas informações de contexto, de forma dinâmica. Isso significa que dispositivos IoT são sensíveis a contexto, tornando mais que necessário que uma plataforma disponibiliza a capacidade de reagir ao contexto de modo que aplicações possam gerar estímulos com as informações inseridas no passado, no momento e especular sobre o futuro a fim de tomar decisões.

6.4.5. ESCALABILIDADE

Um *middleware* para CI deve conter em sua arquitetura a capacidade de escalabilidade entre diversos dispositivos conectados ao ambiente, o maior desafio desta norma é atender é disponibilizar de forma satisfatórias os recursos computacionais para atender a demanda sem prejudicar o funcionamento.

Com a crescente quantidade de dispositivos capazes de captarem informações de contexto em um meio, plataformas optam por estratégias de disponibilização de recursos por demanda.

Isto significa que os recursos serão entregues aqueles dispositivos mais estão necessitando, desta forma um *middleware* consegue trabalhar de forma satisfatórias com o meio sem prejudicar o todo, possibilitando assim que mais dispositivos possam agregar-se a rede e escalar com o objetivo do sistema.

6.4.6. TRATAMENTO DE GRANDES VOLUMES DE DADOS

Devido ao crescente número de dispositivos, geram consequências como a criação de dados de forma gigantesca, seja armazenado em banco de dados ou trafegando na rede.

Assim torna-se mais uma missão a plataforma planejar a forma de trabalho com grandes volumes de dados.

Além disso, uma das consequências, de se obter muitos dados, são os problemas de consultas (bastante conhecidas pela área de computação), este problema pode ser um afunilamento da aplicação, pois é uma das operações mais utilizadas por todas aplicações. Devido a isto, o desafio para gerir os dados de forma que a indexação torne as consultas extremamente ágeis é objetivo a ser conquistado por aqueles que pretendem desenvolver um *middleware* para CI.

6.4.7. SEGURANÇA

Acreditamos que segurança dos dados é dos maiores critérios para uma plataforma, pois nem sempre dispositivos IoT estarão se comunicando por meio de redes seguras, normalmente essas redes são de terceiros e de baixa qualidade.

Fornecer uma segurança de qualidade é dever de um *middleware* para que possa ser preservada a integridade de um sistema, privacidade e consistências dos dados.

6.4.8. GERENCIAMENTO DE DADOS

A forma como as aplicações irão acessar os dados, registrar ou alterar é uma das obrigações de um *middleware*. Sendo assim ele é responsável por gerir as informações de contexto, padronizando a forma acesso das aplicações.

6.4.9. FERRAMENTAS PARA DESENVOLVIMENTO DE APLICAÇÕES

Como já foi dito anteriormente, o real objetivo de uma plataforma é auxiliar no desenvolvimento de novas aplicações, para isto ela deve conter ferramentas que disponibilizem este auxílio.

Essas ferramentas podem ser por meio de API's, ou de forma mais acessível, através de painéis e até mesmo *softwares* que podem dar meio de desenvolvimento para pessoas que não possuem possibilidades de programação. Isto pode ser feito por interfaces disponibilizadas pelo próprio *middleware.*

6.5. FIWARE

Com o objetivo de guiar o futuro da internet (Future Internet - FI), a comissão Europeia aceitou um dos maiores desafios da atualidade (criar um middleware que atenda as demandas de uma CI) com o objetivo de criar uma plataforma que guiasse a FI, daí surgiu o FIWARE (Future Internet Ware) (Batista et al, 2016).

O FIWARE é uma plataforma mediadora (*middleware*) desenvolvida pela União Europeia. Ela contém o objetivo de auxiliar a criação de aplicações para Internet das Coisas (do inglês, *Internet of Things*, IoT). A plataforma tem uma política de código aberto (*open source*) para encorajar novos provedores de módulos (FIWARE, 2016).

Mesmo sendo uma tecnologia recente e em ramo novo para o mercado (dados abertos, dados volumosos e IoT), esse *middleware* já possui servidores em boa parte do mundo, inclusive no Brasil (São Paulo) (FIWARE Lab Node, 2017). Tais servidores (nós) têm como objetivo disponibilizar um ambiente de teste para aplicações, além de armazenar aplicativos para disponibilizar ao mundo todo.

Para tornar a plataforma robusta e de fácil compreensão, o FIWARE divide-se em sete (7) componentes, interoperáveis, que dão suporte tecnológico em diferentes áreas de mercados:

- **Orion Context Broker**: permite a modelagem dos dados de armazenamento e gerenciamento de dados em larga escala, possibilitando aplicações sensíveis ao contexto;
- **Conexão à Internet das Coisas**: agentes (IDAS - Internet das Coisas, IoT) capazes de reunir dados de sensores ou que atuem sobre objetos físicos;
- **Autorização de manuseio (*Handing Authorization*) e controle de acesso às API's (*Access control* to API's)**: contém poderosas estruturas para configuração de acesso e autorização, baseadas em padrões de segurança como: OAuth e *eXtensible Access Control Markup Language* (XACML);
- **Publicação de informações de contexto como dados abertos**: o FIWARE incorpora o CKAN como parte de sua arquitetura. O CKAN é um portal de dados abertos, que permite o gerenciamento dos dados, como também os direitos de uso e quem publicou tais dados. O CKAN, é um portal já difundido e bem aceito pela comunidade como ferramenta para dados abertos, temos exemplo disso no próprio Brasil, estados que usam essa ferramenta para disponibilizar a população os dados armazenados no estado, como por exemplo o estado de Alagoas.
- **Análise de grandes dados de informações de contexto histórico**: incorpora o *software* Cygnuscomo parte da arquitetura, assim como o

CKAN, que permite armazenar dados históricos de informações de contexto, permitindo análise de dados (Big Data) ou consultas avançadas sobre as informações históricas.

- **Criando painéis de aplicação**: o FIWARE disponibiliza nos seus laboratórios a ferramenta Wirecloud, ela foi feita para usuários finais que não possui habilidades de programação, o Wirecloud possibilita a criação de painéis de aplicativos completos a partir de *widgets*.
- **Processamento em tempo real de fluxos de mídia**: incorporando também o Kurento que permite o processamento de multimídia para incorporar recursos de detecção baseado em mídias como fotos, áudios e vídeos. Como por exemplo analisar se em uma foto tirada no seu celular contém o rosto de uma pessoa dos seus contatos.

Além disso, a plataforma disponibiliza um catálogo que contém uma biblioteca de *plug-ins*, com intuito de disponibilizar aos desenvolvedores ferramentas padronizadas, estruturadas e inteligentes para tornar o desenvolvimento de novos aplicativos mais ágil e fácil. O catálogo subdivide-se em dois, habilitadores genéricos (*Generic Enablers* - GE) e habilitadores de domínio específico (*Domain Specific Enablers* - DSE) (FIWARE Catalogue, 2015).

Os GE's fornecem funcionalidades de âmbito geral em ramos da computação, como por exemplo: gestão de dados/contexto, segurança, Internet das coisas, dentre outros. Isto é feito por meio de API's bem definidas e de código aberto. Segundo o FIWARE uma GE deve documentação necessária para que usuários possam construir aplicações capazes de usar mais de uma GE de forma interpelável.

Assim como as GE's, as DSE's também disponibilizam funções. No entanto, elas são úteis para aplicações voltadas para ramos específicos de uma cidade como por exemplo: saúde, energia, dentre outros.

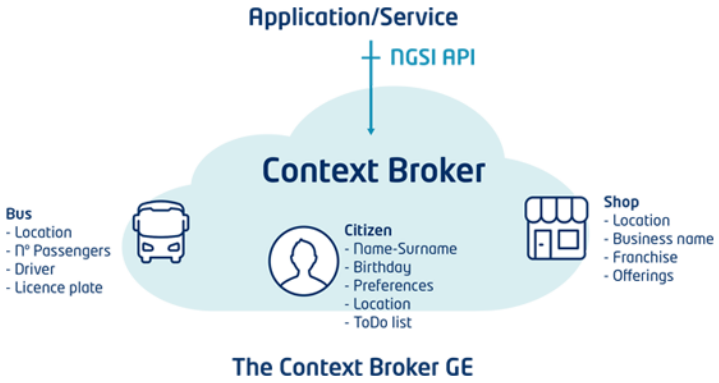
Visto que o FIWARE é um *middleware* abrangente, focamos nossos esforços a desvendar como utilizar um dos componentes do FIWARE, o Orion Context Broker (também chamado somente por Orion), visto que o Orion contém recursos para armazenar grandes quantidade de dados (o que é de extrema importância para o desenvolvimento de nossa aplicação - *Aedes Tracker*). Mostraremos assim como instala, como funciona e como usar esse componente em qualquer aplicação, em especial na nossa.

6.5.1. ORION

O Orion é a base da plataforma FIWARE. É por meio dele que as GE's conseguem se comunicar com o banco de dados. Tudo é feito por meio de

requisições e solicitações (FIWARE - *Tour Guide*, sem data), ou seja, uma aplicação faz uma requisição ao Context Broker, e o mesmo responde a requisição.

Figura 1 - Context Broker

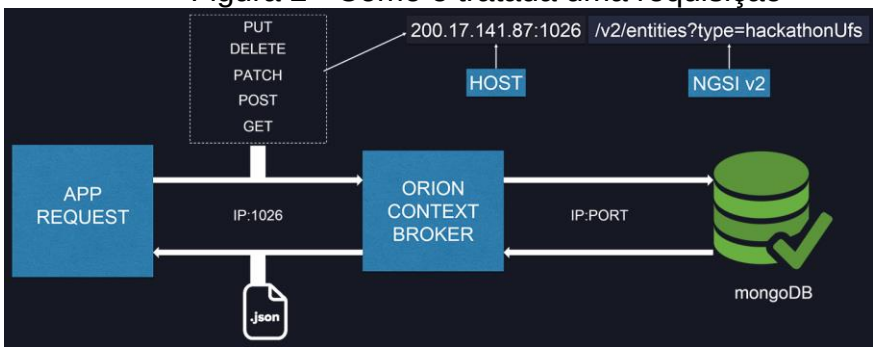


Fonte: FIWARE - Guia quick FIWARE tour

Assim como mostra a figura acima, as requisições podem vir de várias aplicações (*Application/Service*) e em qualquer lugar, desde que tenha uma conexão com a Internet.

Essa comunicação é feita por meio de URL's, as aplicações necessitam informar o IP e porta reservada para o Orion, além disso a URL necessita conter padrões para que o Context Broker saiba como responder as requisições. Esses padrões (desenvolvidos pela própria FIWARE) são chamados NGSI. Tais requisições devem conter cabeçalhos específicos da operação desejada e métodos para que o Orion saiba como deve tratar a requisição.

Figura 2 - Como é tratada uma requisição



Fonte: autoria própria

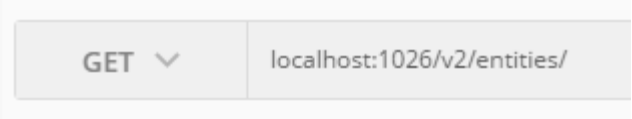
Na Figura 2 é mostrado o fluxo de uma requisição feita por um aplicativo qualquer. Nela o aplicativo define o método, que pode ser put, delete, patch, post ou get (cada método contém um significado para a API) e informa a URL que será enviado os dados da requisição. Esta requisição chega até o Orion, que por sua vez traduz o NGSI v2 e solicita ao banco de dados mongoDB. A resposta do banco é recebida pelo Orion que converte os dados do banco para um arquivo JSON. Por fim, este arquivo é passado como resposta a requisição feita pelo aplicativo.

6.5.2. NGSI v2

Atualmente existem duas versões, NGSI v1 (em desuso e futuramente obsoleta) e NGSI v2 (FIWARE - *Tour Guide*, sem data). A diferença entre elas é que a versão um faz apenas requisições do tipo GET, enquanto a versão dois consegue realizar todos os tipos de métodos em suas requisições. Além disso, toda solicitação necessita passar ou receber dados, que por padrão é usado o formato JSON, um formato leve de transferência de dados.

Uma URL no formato NGSI deve conter o seguinte: método, cabeçalho (s), servidor, porta, versão NGSI e gatilhos.

Figura 3 - Requisição de todos os Dados

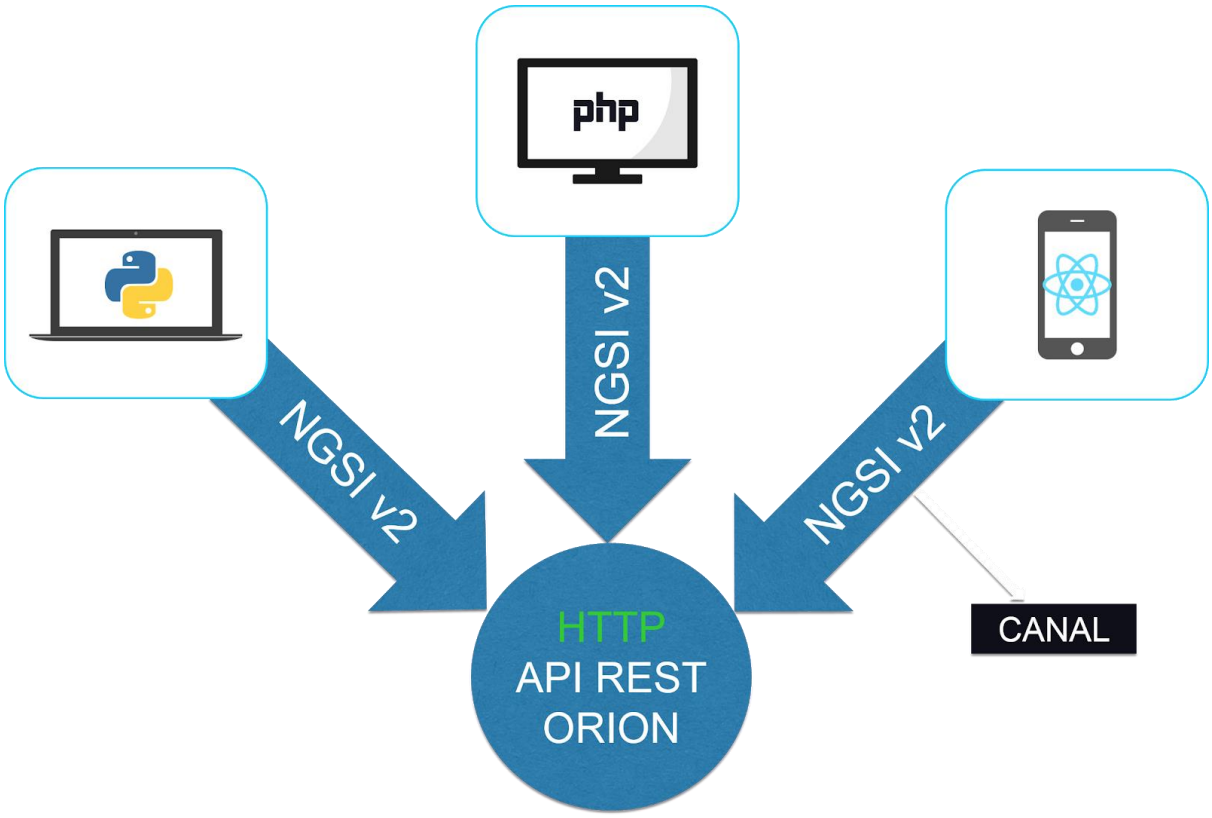


Fonte: autoria própria.

Como por exemplo na Figura 3 foi feita uma requisição do tipo GET, veja que o servidor é localhost e a porta é 1026, informamos também v2 pois queremos usar NGSI v2, logo após passamos o gatilho entities que será recebido pelo orion e nos retornará tudo que estiver no banco de dados em formato JSON. Desta forma o padrão NGSI v2 define a forma como as requisições devem ser feitas ao Orion, possibilitando assim uma escalabilidade grande para criação de novas formas de consulta através da API Orion.

A vantagem de usar essa forma de comunicação é que a tecnologia não fica presa a uma única linguagem, ou seja, o desenvolvedor só precisa se preocupar em saber se a linguagem que está utilizando para desenvolver sua aplicação contém funcionalidades para requisições HTTP e suporta o formato JSON.

Figura 4 - NGSI v2 canal de comunicação



Fonte: autoria própria.

Note que, na figura 4, as aplicações não se comunicam diretamente com o banco, o que traria diversos outros problemas (como por exemplo: desenvolver módulos de segurança, para a comunicação, em cada uma das tecnologias usadas em sua aplicação). A NGSI v2 já é bastante poderosa e possibilita diversos gatilhos e formas de busca ao banco, detalhamos mais sobre esse padrão no tutorial que se encontra no Anexo II.

7. METODOLOGIA OU DESCRIÇÃO TÉCNICA

Nos primeiros meses foram feitas revisões bibliográficas onde buscamos nos aprofundar sobre o conceito *Smart City* e estudar o que está sendo feito na área, buscando também saber qual a visão de uma cidade inteligente sobre a área de saúde.

O estudo sobre FIWARE foi o foco da nossa iniciação científica em seus seis primeiros meses e nós tivemos o objetivo de destrinchar e aprender a desenvolver aplicações que utilizem a plataforma FIWARE. Além disso fomos convidados a participar do grupo SMART UFS, onde seu objetivo também é conhecer mais o FIWARE.

Neste grupo tivemos a oportunidade de ministrar seminários para os integrantes, onde, através do mesmo, conhecemos integrantes familiarizados com a plataforma que forneceram mini-cursos sobre o ambiente de cidades inteligentes e a plataforma FIWARE, tudo através da orientação da orientadora Dr. Leila Maciel de Almeida e Silva que realizou reuniões quinzenais conosco.

Após destrinchar a plataforma buscamos descobrir como instalar e configurar o FIWARE para desenvolver nossa aplicação. Além disso realizamos consultoria com a Dra. Roseli La Corte dos Santos para elaborarmos uma aplicação de monitoramento do mosquito Aedes Aegypti.

Tudo ocorreu conforme o cronograma, de forma objetiva e clara.

CRONOGRAMA DE ATIVIDADES

Atividade	2017 Ago	Set	Out	Nov	Dez	2018 Jan	Fev	Mar	Abr	Mai	Jun	Jul
1. REALIZAR UMA REVISÃO BIBLIOGRÁFICA NA ÁREA DE CIDADES INTELIGENTES, EM ESPECIAL NA ÁREA DE SAÚDE COLETIVA;												
2. ESTUDAR AMPLAMENTE A PLATAFORMA FIWARE;												
3. DESENVOLVER E TESTAR UMA APLICAÇÃO NA PLATAFORMA PARA AUXILIAR A NOTIFICAÇÃO DE FOCOS DO AEDES AEGYPTI;												
4. ESCREVER RELATÓRIOS DE PESQUISA												
5. ESCREVER ARTIGOS CIENTÍFICOS												
6. MINISTRAR SEMINÁRIOS DE PESQUISA.												

8. RESULTADOS PRELIMINARES

O primeiro resultado da nossa IC foi o estudo aprofundado do FIWARE, dado que é uma plataforma muito recente para Cidades Inteligentes. O segundo resultado foi o desenvolvimento da aplicação usando o FIWARE.

Tendo em vista falta ou má coleta de dados relacionados aos mosquitos Aedes Aegypti, procuramos elaborar uma aplicação que consiga coletar de forma fácil e útil para futuros levantamentos de pesquisas e relatórios além da melhor visualização dos dados.

Sendo assim dividimos a aplicação *Aedes Tracker* em dois módulos, módulo mobile e módulo web, para facilitar os estudos e desenvolvimento da aplicação. Com isto, especificarei para o leitor o módulo *Web*.

Este módulo, na verdade, é o responsável por controlar a operação de levantamento de dados sobre o mosquito e visualizar as informações, através de gráficos, mapas e filtros.

Traduzindo para outras palavras, o módulo web é o cérebro da aplicação, enquanto o mobile é o coração, pois é mobile que irá de fato coletar os dados necessários para que aplicação consiga trabalhar as informações e também é o mobile que reformulará a metodologia usada para registrar focos do mosquito.

O módulo mobile foi desenvolvido pelo outro aluno de IC deste projeto.

8.1. FUNCIONALIDADES

Tendo em vista que a gestão e controle do combate ao mosquito é complexa, dividimos a aplicação *Web* com três papéis hierárquico, cada um com funcionalidades chave para o processo de coleta e supervisão de dados, abaixo discutimos especificamente sobre cada um dos papéis e suas respectivas funcionalidades de acordo com as responsabilidades.

8.1.1. COORDENADOR

1. Definir os quarteirões de pesquisa; através da aplicação o coordenador poderá delimitar pontos no mapa, os quais delimitam um quarteirão de pesquisa;
2. Definir o(s) supervisor(es) de cada quarteirão. Este processo não impede que um supervisor seja responsável por mais de um quarteirão;
3. Define a equipe de agentes de um determinado quarteirão, associando assim uma equipe a um supervisor. Assim como os supervisores, uma equipe ou até mesmo um membro de uma equipe pode ficar responsável por mais de uma região de pesquisa;
4. Delimita os laboratórios responsáveis pela região de pesquisa (também pode ser atribuído o mesmo laboratório a mais de uma região);
5. Registra os laboratórios;
6. Registra os supervisores;
7. Registra os agentes de saúde.

8.1.2. SUPERVISOR

1. Cadastra rotas de trabalhos, assim como os quarteirões, um supervisor pode delimitar pontos no mapa para registrar rotas a fim de cobrir todas as residências de uma região de pesquisa. A diferença dessa função para a

função do supervisor (no ponto de vista de desenvolvimento) é que o último ponto registrado no mapa não coincide com o primeiro, formando um caminho e não um ciclo;
2. Atribui rotas a um determinado agente de saúde da equipe responsável pelo quarteirão da rota. Um agente pode ser atribuído a mais de uma rota, ficando a cargo do supervisor definir a quantidade de trabalho por agente;
3. A aplicação disponibiliza ao supervisor uma tela de rendimento dos agentes, nesta tela é informado quantas residências o agente já atendeu e a quantidade de atendimento do dia ou de um dia específico.
4. Conferir os boletins de campo e laboratório (BCL), esse processo deve ser feito junto ao agente de saúde responsável pelo BCL que deverá com o aplicativo mobile em sua mão na tela da vistoria para que o supervisor possa conferir se não houve nenhum erro no processo realizado pelo agente;
5. Registrar BCL, caso o agente de saúde esteja impossibilitado de registrar através do celular. Esse procedimento só pode ser realizado pelo supervisor em casos justificados e o agente deve conter as informações devidamente preenchida em formulários físicos (papel) ou outros meios de registros.

8.1.2. LABORATORISTA

1. Atribuir amostras visualizadas no mapa a sí para análise;
2. Confirmar ou não suspeita de foco.

8.2. TELA DO SISTEMA

Todos os usuários terão acesso a tela de login, onde eles informarão usuário e senha, o sistema ao ler as informações definirá o nível de acesso do usuário e o levará a tela correspondente a sua hierarquia de acesso.

8.2.1. DEFINIÇÃO DAS TELAS DO COORDENADOR

Inicialmente o usuário encontrara-se na tela de entrada, a qual contém um mapa segmentado pelos quarteirões e as residências as quais já foram realizadas as vistorias pelos agentes de saúde.

As residências serão representadas por ícones, assim como mostra a figura abaixo, os quais contêm cores que representam a situação, (i) verde representa que a residência passou por uma vistoria e não foi encontrado foco do mosquito; (ii) amarelo, significa que a vistoria foi rejeitada ou que a vistoria foi realizada mas

encontra-se em aguardo da resposta do laboratório; (iii) vermelho, este é o cenário onde a casa foi passada pela vistoria, foi recolhida amostrar e constatado foco do mosquito Aedes Aegypti pelo laboratório.

Figura 5 - Estados de vistoria

Fonte: direito de uso pessoal (arte feita sob medida).

Além disso nesta tela conterá uma opção de filtro (na parte superior da tela), para buscar os pontos de vistoria de forma filtrada e um menu com as opções de funcionalidade do coordenador à esquerda.

8.2.2. DEFINIÇÃO DAS TELAS DO SUPERVISOR

Assim como o coordenador, o supervisor ao realizar login será levado a uma tela com um mapa, no entanto neste mapa exibirá somente os quarteirões pelos quais o supervisor é responsável e os pontos que já foram realizados vistorias.

Nesta mesma tela o supervisor terá três opções:

1. filtro, nesta tela conterá a mesma forma de filtragem do coordenador, no entanto este filtro busca somente nas regiões de responsabilidades do supervisor;
2. À esquerda existirá um menu com as outras opções de funcionalidade descritas anteriormente;
3. À direita exibirá todos os agentes de saúde ao qual é de inteira responsabilidade do supervisor, essa parte da tela conterá uma forma de busca e ao clicar em um dos agentes o sistema exibirá, em forma de modal, as informações do agente como por exemplo: quantas casas ele vistoriou no dia, todas as vistorias separadas por dia, média de vistoria, meta, etc.

8.2.3. DEFINIÇÃO DAS TELAS DO LABORATORISTA

Ao efetuar o login o laboratorista verá um mapa em tela cheia com todos os pontos que são responsáveis pelo laboratório, em que ele encontra-se vinculado, diferente dos outros pontos este não contém diferenciação de cores, pois os

mesmo servirão apenas para indicar que existem amostras relacionados a uma determinada residência que necessitam serem analisadas pelo laboratorista.

Neste ponto do sistema o usuário poderá clicar nas residências exibidas no mapa e atribuir a si as amostras da vistoria.

Após realizar o processo de atribuição o sistema exibirá a direita a lista de amostra que estão sob responsabilidade do usuário, essa lista conterá duas opções, confirmar foco do mosquito ou informar que não há foco, ambas as telas levam para um painel de um modal ao qual terá informações que deverão ser preenchidas pelo laboratorista.

8.3. FORMULÁRIOS

8.3.1. BOLETIM DE CAMPO E LABORATÓRIO (BCL)

Apesar deste formulário fazer mais sentido na aplicações mobile, parte deste formulário é preenchido na aplicação web, além de que, o supervisor pode registrar esse boletim em casos que o agente não consiga. Por isso, faz-se necessário descrever os dados do formulário.

Este formulário contém todas as informações necessárias que devem ser preenchidas em campo pelos agentes de saúde, com informações que irão preencher indicadores e terá um papel fundamental para decisões de combate ao mosquito. Segue abaixo os dados que deverão ser preenchidos para o BCL:

- **Número do quarteirão**: esse número é representado pelo identificados (ID) do sistema, tal informação é coletada automaticamente;
- **Endereço**: ao coletar longitude e latitude pelo aplicativo mobile, o sistema recolherá outras informações como rua, avenida, praça, etc, através da API Maps do google.
- **Número**: esta informação deverá ser informada pelo agente;
- **Complemento**: informar outros detalhes da localização como por exemplo: condomínio (juntamente com bloco e apartamento), pontos de referências, cor da casa, etc.
- **TB**: marcar um check box quando o imóvel for um terreno baldio.
- **Situação**: existem três tipos de situação possível, (i) fechado, quando o imóvel encontra-se fechado, (ii) recusado, quando os responsáveis recursão-se a receber os vistoriadores e (iii) trabalhado, quando a atividade ocorreu sem nenhum problema;
- **Amostra**: o usuário registra uma amostra coletada em uma vistoria, ao registrar a amostra o usuário deve informar de onde foi coletada a amostra, por exemplo: de caixas d'água, pneu, jarro de planta, etc. Sendo assim uma

única vistoria conterá uma lista de amostra. Além disso uma amostra contém campos adicionais que serão preenchidos somentes por laboratorista, que são eles: número Aedes Aegypti, número de larvas, número de pupas e data (esse campo é pego automaticamente após registro do laboratorista).

9. CONCLUSÃO

O objetivo dos nossos estudos foi adquirir conhecimentos sobre o FIWARE e aprender como utilizar seus recursos em uma aplicação real. Para isto, tivemos que acessar a documentação direta do FIWARE, pois esta ferramenta é nova e com poucos artigos e guias de como usá-la.

Neste processo de aprendizado desvendamos que estas duas formas de usar o FIWARE, a primeira é solicitar uma máquina direto do FIWARE Lab, a qual necessita informar o objetivo do projeto para aprovação ou baixar o componente do FIWARE (no nosso caso FIWARE Orion), ensinado como instalar no Anexo I.

Além do problema de documentação, tivemos também problemas de contato com ministério de saúde, Dra. Leila nossa orientadora tentou se comunicar com a secretaria através da UFS de modo formal, ao qual não obteve sucesso. Sendo assim foi tentado o contato direto através de uma representante do ministério e da mesma forma não obtivemos sucesso pois foi demonstrado através da representante que o ministério de saúde não tinha interesse em ajudar a planejar a aplicação junto a nós.

Sendo assim passamos cerca de três meses improdutivos pois necessitamos de um auxílio para elaborar nossa aplicação.

Mesmo com todos todos os problemas tentamos um terceiro contato, a Dra. Roseli, esta que nos recebeu com informações concretas para elaborar nossa aplicação.

Sendo assim conseguimos definir funções dos agentes de saúdes, formulários necessários para levantamento de relatórios e elaboração da bases de dados.

10. REFERÊNCIAS BIBLIOGRÁFICAS

Bayer Jovens, Be-a-bá para entender as CIdades Inteligentes. Disponível em: <https://www.bayerjovens.com.br/pt/colunas/coluna/?materia=be-a-ba-para-entender-as-cidades-inteligentes/>. Acesso em 24 de janeiro de 2018.

BECODE - O que é API? REST e RESTful? Conheça as definições e diferenças. Disponível em: <https://becode.com.br/o-que-e-api-rest-e-restful/>. Acesso em 09 de junho de 2018.

Combate Aedes, casos de dengue no Brasil caem 90% em 2017. Disponível em: <http://combateaedes.saude.gov.br/pt/noticias/908-casos-de-dengue-no-brasil-caem-90-em-2017>. Acesso em 15 de janeiro de 2018.

CORS - Entendendo o CORS - parte 1. Disponível em: <https://medium.com/@alexandremjacques/entendendo-o-cors-parte-8331d0a777e1>. Acesso em 25 de junho de 2018.

CYGNUS - Cygnus. Disponível em: <http://fiware-cygnus.readthedocs.io/en/latest/>. Acesso em 26 de junho de 2018.

DATASUS - Ministério de Saúde, e-SUS AB Território. Disponível em: <https://play.google.com/store/apps/details?id=br.gov.saude.acs&hl=pt>. Acesso em 13 de janeiro de 2018.

DATASUS - Ministério de Saúde, Viva Bem. Disponível em: <https://play.google.com/store/apps/details?id=br.gov.datasus.vivabem>. Acesso em 13 de janeiro de 2018.

Department of Economic and Social Affairs, The World at Six Billion. Disponível em: <http://www.un.org/esa/population/publications/sixbillion/sixbillion.htm>. Acesso em 13 de janeiro de 2018.

DEPURAÇÃO - Depuração (Debug ou debugging). Disponível em: <https://pt.wikipedia.org/wiki/Depura%C3%A7%C3%A3o>. Acesso em 25 de junho de 2018.

Docker Inc. - docs, overview of docker compose. Disponível em: <https://docs.docker.com/compose/overview/#development-environments>. Acesso em 21 de janeiro de 2018.

Docker Inc., what is docker. Disponível em: <https://www.docker.com/what-docker>. Acesso em 21 de janeiro de 2018.

FIWARE, repositório docker. Disponível em: <https://hub.docker.com/r/fiware/orion/>. Acesso em 05 de janeiro de 2018.

FIWARE Catalogue, the fiware catalogue. Disponível em: <https://catalogue.fiware.org/>. Acesso em 22 de janeiro de 2018.

FIWARE Developers, start using FIWARE right now. Disponível em: <https://www.fiware.org/developers-entrepreneurs/>. Acesso em 24 de janeiro de 2018.

FIWARE Lab Node, FIWARE lab node. Disponível em: <http://infographic.lab.fiware.org/>. Acesso em 24 de janeiro de 2018.

FIWARE-ORION - Api Walkthrough (v2). Disponível em: <https://fiware-orion.readthedocs.io/en/master/user/walkthrough_apiv2/index.html>. Acesso em 17 de janeiro de 2018.

FIWARE - Tour Guide, development of context-aware applications. Disponível em: <http://fiwaretourguide.readthedocs.io/en/latest/development-context-aware-applications/introduction/>. Acesso em 02 de janeiro de 2018.

Hackathon Carmelita, maratona cívica de programação e inovação. Disponível em: <https://sites.google.com/dcomp.ufs.br/hackathon-carmelita>. Acesso em 17 de janeiro de 2018.

HTML - HTML. Disponível em: <https://pt.wikipedia.org/wiki/HTML>. Acesso em 26 de junho de 2018.

HTTP 1.1 - Os métodos HTTP: quais são e para que servem. Disponível em: <http://gabsferreira.com/os-metodos-http-e-a-diferenca-entre-eles/>. Acesso em 22 de junho de 2018.

INTEROPERABILIDADE - O que é Interoperabilidade. Disponível em: <https://www.diegomacedo.com.br/o-que-e-interoperabilidade/>. Acesso em 26 de junho de 2018.

JAVASCRIPT - JavaScript. Disponível em: <https://pt.wikipedia.org/wiki/JavaScript>. Acesso em 26 de junho de 2018.

LOOPBACK - Laço de volta (Loopback). Disponível em: <https://pt.wikipedia.org/wiki/Loopback>. Acesso em 25 de junho de 2018.

OAuth - OAuth. Disponível em: <https://pt.wikipedia.org/wiki/OAuth>. Acesso em 25 de junho de 2018.

PHP - O que é o PHP. Disponível em: <http://php.net/manual/pt_BR/intro-whatis.php>. Acesso em 28 de junho de 2018.

Requisitos e Plataformas de *Middleware* para Cidades Inteligentes - Universidade Federal do Rio Grande do Norte Instituto Metrópole Digital. p. 47, 2016.

Same-Origin Policy - Same-Origin Policy. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy>. Acesso em 25 de junho de 2018.

Silvano Vilela, Aplicativo e-saúde mostra seu cartão nacional de saúde SUS, consultas, medicamentos. Disponível em: <https://www.plugbr.net/aplicativo-e-saude-mostra-cartao-nacional-saude-sus-consultas-medicamentos-como-baixar-acessar-app/>. Acesso em 13 de janeiro de 18.

TUTORIALS GETTING-STARTED (GIT HUB) - Fiware. Disponível em: <https://github.com/Fiware/tutorials.Getting-Started/blob/master/README.md>. Acesso em 20 de maio de 2018.

TRACE - Método HTTP Trace. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods/TRACE>. Acesso em 25 de junho de 2018.

Urbe Lab, em 30 anos, a população urbana mundial deve ultrapassar as 6 bilhões de pessoas. Disponível em: <https://urbe.me/lab/em-30-anos-a-populacao-urbana-mundial-deve-ultrapassar-as-6-mil-milhoes-de-pessoas/>. Acesso em 13 de janeiro de 2018.

URL - Significado de URL. Disponível em: <https://www.significados.com.br/url/>. Acesso em 25 de junho de 2018.

XACML - eXtensible Access Control Markup Language. Disponível em: <https://en.wikipedia.org/wiki/XACML>. Acesso em 25 de junho de 2018.

11. ANEXO

Dado que a documentação do FIWARE é dispensa em vários documentos e pouco didática em alguns casos, o objetivo deste apêndice é ensinar como instalar

e configurar a plataforma, conhecimento ao qual desperdiçamos muito tempo para adquirirmos.

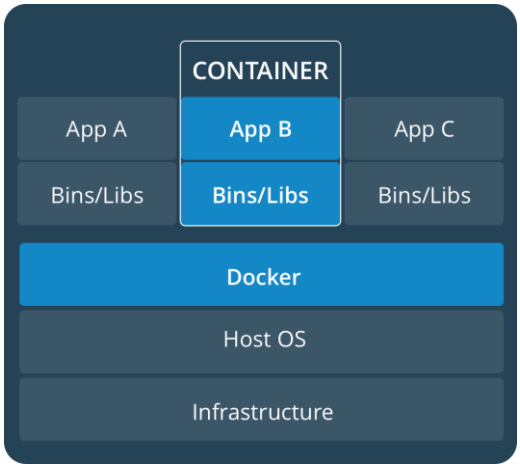
11.1. ANEXO I - INSTALANDO E CONFIGURANDO O ORION

Existem duas formas para instalar o Orion localmente, uma delas é instalá-lo como um software, ou seja, instalando todas dependências na máquina e configurar para que tudo ocorra normalmente ou instalar através do docker como um container. Sendo assim definimos utilizar o docker, pois o mesmo abstrai toda a problemática de configuração.

11.1.1. DOCKER

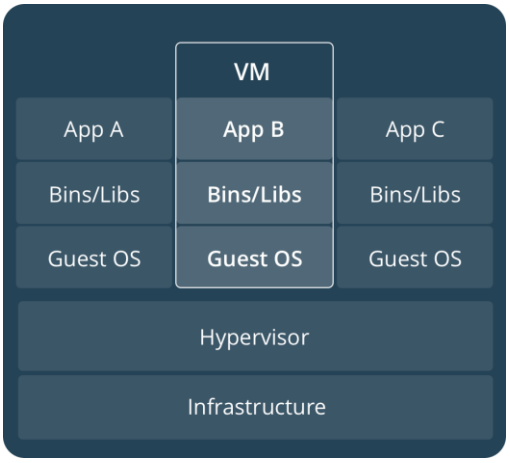
O Docker é um sistema de virtualização de máquina baseado em containers (Docker Inc. - what is docker, 2018). Enquanto um virtualizador tradicional é baseado em virtualização de máquinas e necessita de um sistema operacional (SO) para cada virtualização, o que acarreta em vários problemas para o servidor como por exemplo: gerenciar o tempo de licença de cada SO instalado nas máquinas virtuais. O Docker é baseado em um único SO, ou seja, o usuário instala o docker em uma única máquina e cria seus contêineres, onde estará instalado tudo que for necessário para rodar a aplicação do usuário.

Figura 1 – Container



Fonte: Docker - What-Container

Figura 2 - Máquina Virtual



Fonte: Docker - What-Container

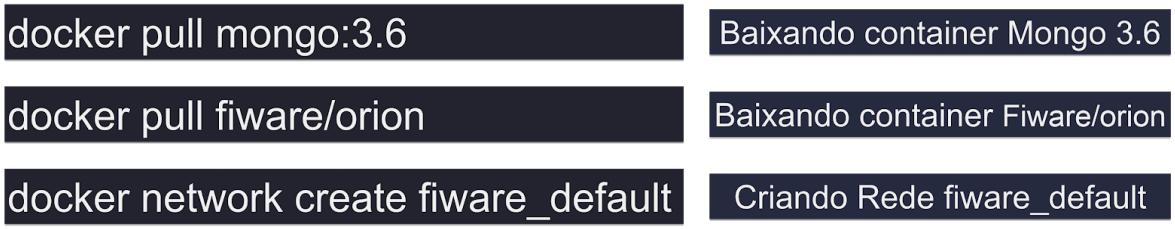
Note que o docker está instalado acima de um SO e que seus contêineres não necessitam de outro SO para rodar as aplicações do usuário, pois tudo aquilo que é necessário como bibliotecas e plug-ins já estão contidos no container da aplicação. Além disso uma máquina virtual fragmenta recursos do servidor como memória e espaço em disco para que sejam reservados para cada máquina virtual, diferentemente de uma VM o docker faz o gerenciamento dos recursos de forma que os recursos só são disponibilizados para os contêineres quando é necessário evitando então que recursos físicos fiquem reservados sem serem usados.

O docker também provém diversos outros recursos para servidores como por exemplo: *docker compose* e *docker engine swarm*. Duas ferramentas poderosas desse *software*, no entanto, não provêm aos nossos estudos explorar o *docker*, por isso, assumimos que o leitor já possui instalado e configurado o *docker*.

10.1.2. ORION

Para realizar a instalação do componente Orion context broker do FIWARE, é necessário que façamos o *download* dos containeres (mongoDB e fiware-orion) através do *docker*. Para realizar este download execute os seguinte comando:

Figura 3 - Baixando Imagens dos Containeres



Fonte: autoria própria

Feito isto seu docker conterá então duas (2) imagens de referência (mongo:3.6 e fiware/orion) e uma rede interna chamada "*fiware_default*", possibilitando assim a criação do Orion, da seguinte forma:

Figura 4 - Criando e Iniciando o Orion

```
docker run –d –name=mongo-db –network=fiware_default –expose=27017 mongo:3.6 –bind_ip_all --smallfiles
```

Criando e executando container mongo-db

```
docker run –d –name fiware-orion –h orion –network=fiware_default –p 1026:1026 fiware/orion –dbhost mongo-db -corOrigin __ALL
```

Criando e executando container fiware-orion (Com o cabeçalho CORS)

Fonte: autoria própria

No primeiro comando criamos e iniciamos um container do banco de dados mongoDB chamado "*mongo-db*" (trecho de comando: -name=mondo-db) e conectamos esse banco a nossa rede, criada anteriormente, chamada "*fiware_default*" (trecho de comando: -network=fiware_default), essa conexão faz-se necessária para que o Orion consiga enxergar o nosso banco de dados e possa buscar, registrar, alterar e remover informações.

Assim como no primeiro comando, o segundo cria e inicia um container, no entanto, este container é o próprio Orion que chamamos de "*fiware-orion*" e, assim como o "*mongo-db*", conectamos o container na mesma rede "*fiware_default*", no entanto como este é o container que o usuário fará o acesso diretamente precisamos definir a porta de acesso, por definição - do FIWARE - a porta de acesso padrão é a 1026 (trecho de comando: -p 1026:1026), no entanto nada impede que o leitor possa criar o container em outra porta ou mais de uma porta. É válido também ressaltar que durante a criação do Orion Context Broker foi executado um outro comando (trecho de comando: -corsOrigin __ALL) para que não ocorra futuros bloqueios de navegadores nas requisições feitas ao servidor - falaremos melhor sobre o problema CORS no tópico 5.X.

11.2. ANEXO II - USANDO A API ORION

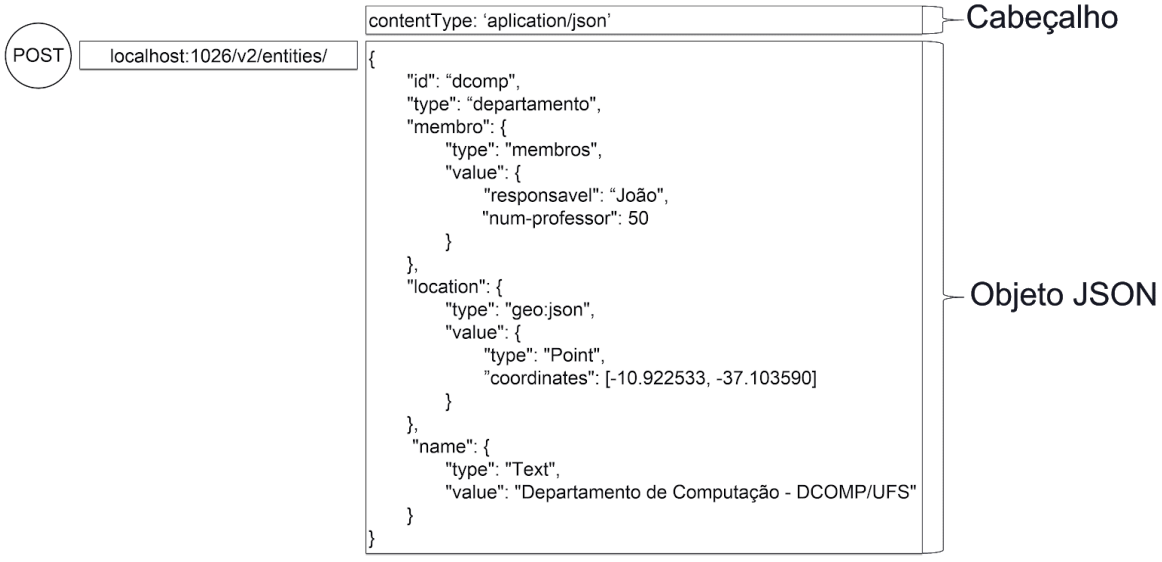
Neste anexo iremos abordar e ensinar como usar o Orion Context Broker, para isso precisamos que leitor já tenha instalado e configurado em sua máquina o Orion, que está descrito o passo-a-passo no Anexo I. O foco principal é abordar, amplamente, os padrões usados no NGSI v2.

Assim como foi dito anteriormente o NGSI é que um canal de comunicação para a nossa base de dados Orion, ou seja, precisamos conhecer os padrões exigidos para que possamos utilizar a ferramenta em nossas aplicações. Esses padrões irão nos ensinar a forma de como requisitar uma inserção de novo registro (POST), recuperar informações (GET), alterar (PUT e PATCH) e como deletar (DELETE) - esses métodos foram descritos no tópico 5.4. HTTP - começaremos então com a inserção.

11.2.1. INSERINDO REGISTROS

Para inserir um novo registro, também chamada de entidade, o Orion exige que o dado JSON contenha dois campos obrigatoriamente, ID e TYPE, o ID deve um campo único, ou seja, é o campo chave de referência para a entidade, já o type é para classificação dos dados, por exemplo: na Universidade Federal de Sergipe (UFS) existem muitos departamentos, todo departamento contém uma sigla única, DCOMP - Departamento de Computação. Neste exemplo o id seria dcomp e o type departamento. Além do id e type, existem os atributos de uma entidade, dando continuidade ao exemplo, um possível atributo seria o responsável pelo departamento, o número de professores do departamento, a localização em latitude e longitude, todos estes são exemplo de atributos. Vamos mostrar então como isso funciona na prática, usando uma ferramenta chamada POSTMAN:

Figura 3 - Inserindo uma entidade



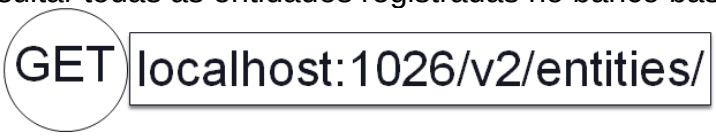
Fonte: autoria própria

11.2.2. CONSULTANDO DADOS

A API Orion foi preparada para receber consultas de diversas formas, sendo assim, versátil para os desenvolvedores desejar, neste exemplo

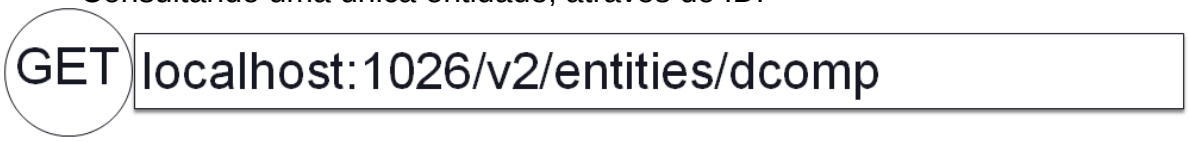
mostraremos as diferentes formas de consultar e explicaremos a resposta que o Orio trás. Por exemplo:

- Para consultar todas as entidades registradas no banco basta fazer:

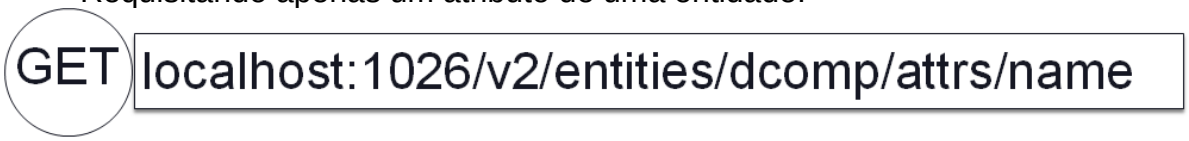


- Filtrar as entidades por tipo:

- Consultando uma única entidade, através do ID:

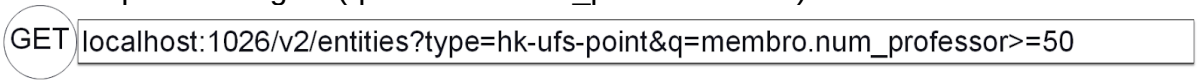


- Requisitando apenas um atributo de uma entidade:



- Consultando atributos específicos de um tipo de elemento no banco:

- Filtrando por comparação de valores nos atributos, para essa consulta usamos o gatilho “q” e informamos qual a variável que iremos usar além da expressão lógica (q=membro.num_professor>=50):



Perceba que ao consultar atributos específicos usamos o parâmetro “options” com o valor “keyValues“, essa opção serve para termos um retorno mais compacto na resposta e nela só existem dois atributos: “keyValues” e “values”. Além disso existem diversas outras formas de fazer requisições a nossa API, o leitor pode saber mais em: fiware-orion e tutorials getting-started (Git Hub), os links encontram-se nas referências.